

# Sensor Based Path Planning in Highly Constrained Environments for Agile Autonomous Vehicles

Kito Berg-Taylor<sup>\*</sup>, Keehong Seo<sup>†</sup> and Soon-Jo Chung<sup>‡</sup>

*Aerospace Engineering Department, Iowa State University, Ames, IA, 50011, USA*

**This paper presents a class of new methods which explore and describe an unstructured environment according to the free space as seen by a vehicle within the environment. The proposed methods use such sensor-based information to present a safe, goal-seeking path through the environment. The methods are suitable to both online, reactive path-finding and obstacle avoidance as well as online or offline global navigation and goal-seeking behaviours. Experimental results on ground vehicles and in simulation are presented to demonstrate the path-planning capabilities of the proposed methods.**

## I. Introduction

THE continual advances being made in sensor technologies have recently given UAVs and especially small, autonomous indoor micro aerial vehicles (MAVs) the potential to be an effective solution for a wide-ranging selection of tasks.<sup>1</sup> The small size and three-dimensional mobility of MAVs gives them significant advantages over ground vehicles and other traditional robots which operate in constrained areas. However, these same capabilities present significant challenges in the development of the robust and effective online path-planners necessary for fully autonomous flight.

While there is a rich body of research related to path-planning in sparse environments for both individual<sup>2</sup> and swarms<sup>3 4 5</sup> of autonomous Unmanned Aerial Vehicles (UAVs), the introduction of similar capabilities in MAVs and UAVs operating in crowded or highly constrained environments has been more limited<sup>6 .7</sup>. The reasons for this are diverse. The small size and payload capacity of most MAVs makes it impossible to carry a wide variety of sensors popular on larger vehicles. In addition, it is often impossible to reap the benefits of sensor fusion on a vehicle that is limited to a single sensor. Path-planning in constrained environments presents a special challenge due to the limits on viable configurations of the vehicle and consequences of poor path-planning.

One sensor that has shown considerable promise in use on MAVs is the camera. The small size and weight of modern CMOS camera modules has allowed single and even stereo cameras to be outfitted to MAVs<sup>8</sup>.<sup>7</sup> Cameras provide a large amount of information about the environment, which is of great benefit when trying to perform online path-planning. Recent improvements in processing of camera images have made planning in 3D from this information feasible.<sup>9</sup>

There have been two major approaches to the problem of path-planning in constrained three-dimensional environments. The first approach is commonly known as quasi- two-dimensional or 2.5 dimensional path-planning, and is similar to constrained two dimensional movement in a plane. However 2.5D path-planning typically takes additional measures to incorporate information from the third dimension so as to provide additional flexibility when it becomes necessary to move out-of-plane. The second major approach is the full three dimensional path planner. These methods provide the maximum amount of robustness in complex

---

<sup>\*</sup>Research Assistant, Aerospace Robotics Laboratory, thekito@gmail.com

<sup>†</sup>Postdoctoral Associate, Aerospace Robotics Laboratory, kseo@iastate.edu

<sup>‡</sup>Assistant Professor, Director of the Aerospace Robotics Laboratory, Department of Aerospace Engineering, sjchung@iastate.edu

scenarios, but come at the cost of additional processing overhead. Increasing the dimensionality of the problem by one can easily cause an order of magnitude increase in the amount of data that has to be processed. This has severely limited the applicability of full three dimensional path planners.

The Aerospace Robotics Lab at Iowa State University has recently begun work on cooperative indoor and terrain masking operations between ground and small aerial vehicles as partially introduced in.<sup>10</sup> As part of this work a need for path planning methods that could provide goal seeking behaviour in environments that are cluttered with obstacles and other constraining factors were needed. These methods were required to be highly responsive to the limitations inherent in the environment, as well as being able to operate solely using the vehicle’s onboard active sensors. Because the environments are often GPS-denied a global position estimate could not be a requirement for safe navigation. In this paper three such path-planners are presented, and as part of the lab’s ongoing research these methods will be further refined and tested.

In this paper, we shall be presenting lightweight methods for exploring an environment and performing goal seeking behaviours constrained within a plane. The Heuristic Circular Sector Expansion (HCSE) method is presented in section IV.A; it provides a capable method of generating a constantly updated path tree which can be used to find shortest-path, safest-path or exploratory routes through an environment. This method has been experimentally validated on the Aerospace Robotics Lab’s ground rovers in dynamic indoor and outdoor environments. The edge cutting tree (ECT) algorithm is presented in section IV.B as another 2D collision-free sensor-based path-planning algorithm. ECT generates multiple paths from a binary tree geometrically constructed from the sensor information. It is designed to be efficient especially for indoor environments with walls and hall ways. The generated paths can be optimized by adjusting the waypoints laterally on the edges that cut through the paths and then can easily be identified if the paths lead to dead-ends or not. Currently, the method is waiting to be experimental validated. We shall then extend HCSE to improve its applicability to aerial vehicles through the use of a 2.5 dimensional Cylindrical Sector Expansion (CySE). This method, presented in section IV.C, provides the ability to incorporate altitude changes into the path planner for better agility in complex environments. Finally, we will show a fully robust method of finding paths in the most complex environments through the expansion of spheres in all three dimensions. Spherical Sector Expansion (SSE) is presented in IV.D, and has been demonstrated in MATLAB simulations to provide complete information about the available paths in an environment.

## II. Related Work

The robotics community has been interested in path-planning and navigation in indoor environments for several decades, although the bulk of research in that area has been limited to ground vehicles. One method used in the robotics community that has found success is the usage of the edges of the Voronoi Diagram (GVD) of an environment to plan a path. This guarantees a path that is not only free of obstacles, but is in fact the safest path through the environment. Ronnback showed that it is possible to quickly find the edges of the Generalized Voronoi Diagram of laserscan data by expanding a series of circles starting from the vehicle location.<sup>11</sup> Ronnback then further expanded this method to provide dynamic path planning and topographical mapping for a self-guided wheelchair.<sup>12</sup>

There are diverse approaches for the 2D path-planning problem. Ours is close to the computational geometry or topology-based approaches such as Ref.13. The method known as rapidly-exploring random tree (RRT<sup>14</sup>) is widely used for the purpose and it has common characteristics as ours in using the tree structure. An advanced method<sup>15</sup> from RRT was introduced by Frazzoli to deal with moving obstacles in the environment efficiently. Such approaches usually have well defined goal to reach and assume that all the information of the environment is given in advance so that the algorithms can compute best paths a priori. Our approach is a little different in that only the information that gathered from each scan of the range finding sensor is used and the planning aims at on-line computation. As a trade-off, our approaches lack the proof of probabilistic completeness of the algorithms but we rather chose to experimentally validate the methods.

The potential for planning safe paths in three dimensional environments was recognized by Vandapel, Kuffner and Amidi,<sup>16</sup> where a grid of “free space bubbles” was used to find a safe path through a forest

canopy. This offline method was performed using laser scan data taken from a prior flight through the forest with a helicopter. By looking at the intersections of the spheres it was possible to determine the path which stayed the greatest distance from obstacles at all times.

### III. Problem Formulation

#### III.A. Movement in a plane

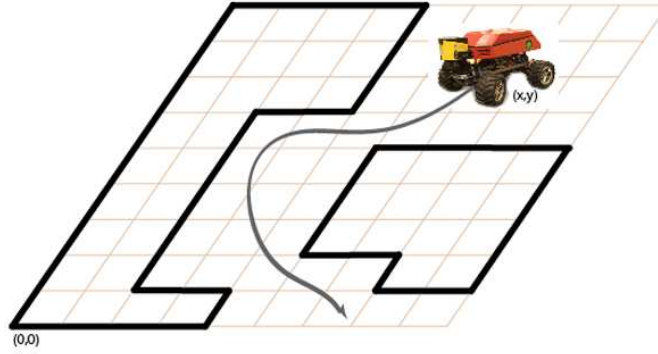


Figure 1. Overview of the strict 2D pathfinding problem

Figure 1 summarizes the problem of vehicle movement in a plane. In the strict 2D case the motion of the vehicle is limited to motion within a plane and all information about the environment that does not lie in the plane is discarded. In this world model the vehicle's state can be described as

$$\hat{x} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix} \quad (1)$$

where  $x$  and  $y$  are the vehicle's position on the plane and  $\psi$  describes its heading relative to some reference heading. The environment itself is then described as a collection of obstacles  $\Lambda$ , such that

$$\Lambda \in \{\lambda | \lambda \rightarrow \mathbb{R}^2\} \quad (2)$$

Alternatively it can also be convenient to represent the world obstacles in polar coordinates centered around the vehicle, where

$$\bar{\Lambda} = \{[d \ \alpha]^T | d \in \mathbb{R}, -\pi \leq \alpha \leq \pi\} \quad (3)$$

In this paper we will be representing the vehicle as a sphere with constant radius  $r_{vehicle}$ . In the 2D case a collision is defined as an obstacle  $\lambda_i$  being within the vehicle's safety circle  $r_{safety}$ .

#### III.B. Quasi-2D navigation

Quasi-2D motion is conceptually similar to movement in a plane. In the 2.5D world model the altitude information in the world is divided into a set of discrete intervals and the vehicle motion is limited to movement

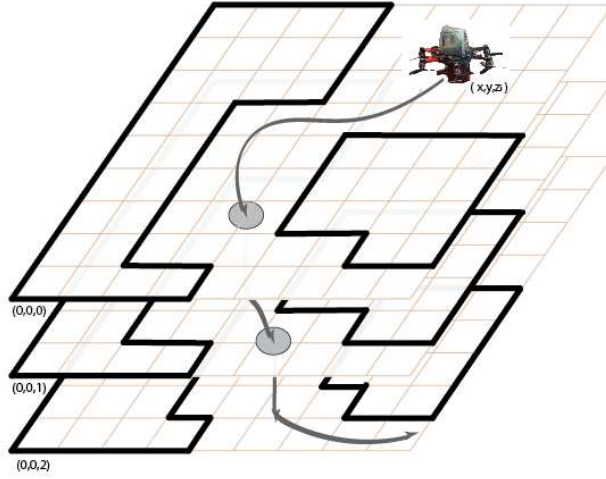


Figure 2. Overview of the quasi 2D pathfinding problem

within the current plane and transitional movements between two planes.

In many environments the motion of an aerial vehicle can be reduced to quasi-2D motion at a constant altitude. This approximation reduces the order of the resulting pathfinding problem. This approximation can be further improved by adding the height information back into the problem through a discretization of the vertical axis. This leads to the 2.5D world approximation. In this approximation the vehicle's state is described by

$$\hat{x} = \begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix} \quad (4)$$

where

$$z \in \{f(x) | x \in \mathbb{I}\} \quad (5)$$

Similarly the obstacles in the environment are binned into discrete layers where all obstacles with an altitude in the range between two discrete values are grouped together. In this environment model a collision is defined as the set of states where an obstacle  $\lambda_i$  has a 2D distance within the safety radius of the vehicle, and an altitude difference from the vehicle less than half the altitude discretization distance.

### III.C. Full 3D navigation

In some situations the limitations of the 2.5D approximations can limit the possible motions of the vehicle. This is especially true of helicopters and other vehicles with operating advantages in constrained environments, which are capable of performing maneuvers which would require full three-dimensional path planning. Figure 3 shows an example scenario where full 3D pathfinding can provide benefits.

In the full 3D pathfinding problem the vehicle's state is

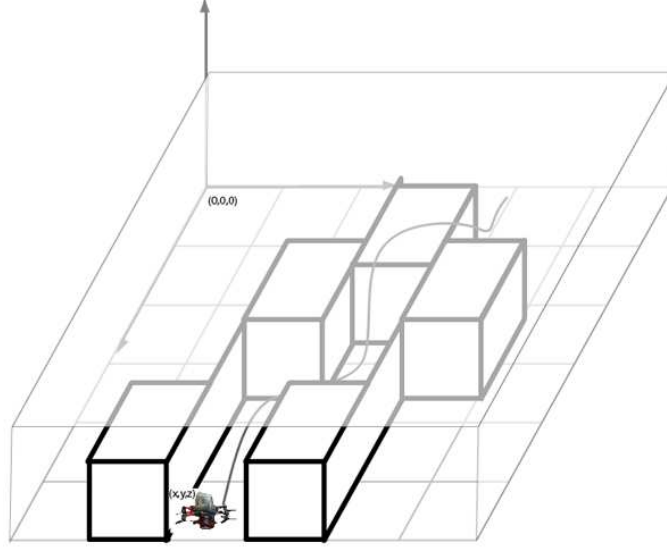


Figure 3. Overview of the full 3D pathfinding problem

$$\hat{x} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} \quad (6)$$

and obstacles in the environments are taken to be the set of points in three dimensional space which are known by the vehicle to be occupied

$$\Lambda \in \{\lambda | \lambda \rightarrow \mathbb{R}^3\} \quad (7)$$

A collision in this environment is defined as an obstacle being present within the vehicle's safety sphere.

## IV. Proposed Method

### IV.A. Heuristic Circular Sector Expansion

Here we present a method for describing the free space in a two dimensional environment. This method gives an approximation to the Generalized Voronoi Diagram (GVD) of the obstacles, and by following the edges of the path the maximum distance from all obstacles can be maintained. This is also known as the safest path.

This method is provided as an alternative to CSE as presented in [ 11] for use in large and structured environments. Heuristic CSE has the advantage of being able to operate not only on point obstacles, but also on geometric features. Here we extract lines from the point obstacle representation of the world to eliminate redundant elements in the world description. We then use this reduced information as part of HCSE to develop safe paths.

#### IV.A.1. Feature Extraction

There exist several recursive methods for locating line features in a cluster of points such as RANSAC<sup>17</sup>. Unfortunately recursive methods have unbounded worst-case computational complexity and are therefore ill-suited to agile vehicles which must have deterministic algorithm performance. Here we suggest a linear time algorithm for extracting lines and corners from laser scan data. The extraction of representative walls and corners from image data is an active area of research<sup>9</sup> and will not be addressed here.

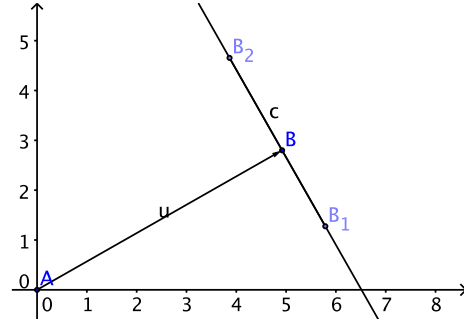


Figure 4. Vector representation of an arbitrary line.

Here we will be representing lines as a vector which carries information about both its location and slope. For the line  $c = \overline{B_1 B_2}$  shown in figure 4 the vector that represents the line is defined as

$$\vec{u} = \left[ (\vec{B_2} - \vec{B_1})^\perp \cdot (\vec{B_1} - \vec{A}) \right] (\vec{B_2} - \vec{B_1})^\perp \quad (8)$$

where  $\vec{x}^\perp$  is the unit vector perpendicular to  $\vec{x}$ . For convenience we will take the point  $A$  to lie on the origin.

Before we can begin to fit lines to the data we will have to identify areas of the data that are discontinuous. These areas arise from occlusions to the line of sight of the sensor. Since a line segment should never extend across a discontinuity it is necessary to first identify the discontinuities. This is done by applying a threshold to the change in distance from one scan point or obstacle to the next. This difference is normalized by the average distance to the obstacles because obstacles further away from the laser rangefinder will naturally be further separated from each other. The distance is defined as

$$r = \frac{2(d_k - d_{k-1})}{\sin(\Delta\alpha)(d_k + d_{k-1})} \quad (9)$$

where  $d_k$  is the distance to the  $k$ th obstacle and  $\Delta\alpha$  is the angular separation between scans. If  $r$  is greater than some threshold value we will consider obstacles  $\lambda^k$  and  $\lambda^{k-1}$  to be the ends of a discontinuity.

Our method of fitting a line to the data involves attempting to approximate the minimum mean-squared error (MMSE) line through the data points. This is done by going through the data and iteratively updating the MMSE line fit. Obstacles which have a perpendicular distance to the MMSE line fit which exceeds some threshold are considered to be corners, and a new line fit is begun from that point. The lines are not allowed to extend across discontinuities because we consider discontinuities to represent the beginning of a new line object.

At each scan point  $\lambda^k$  we begin by drawing a line from the previous line's start point to the scan point two steps forward.

$$l_1^k = \lambda^{k+2} - \lambda_0^{k-1} \quad (10)$$

We then take the perpendicular distance from  $\lambda^k$  to the angle bisector of  $l_1^k$  and the previous best fit line  $l^{k-1}$

$$r = (l_2^k)^\perp \cdot (\lambda^k - \lambda_0^{k-1}) \quad (11)$$

where  $(l_2^k)^\perp$  is a unit vector perpendicular to the bisector line  $l_2^k$ .

If the distance  $d$  is above some threshold then the  $\lambda^k$  is taken to be a corner point and the best fit line is taken to be  $l^{k-2}$ . This line has a start point at  $p_0^{k-2}$  and an end point  $\lambda_1^k$  defined as the closest point to  $p_k$  which lies on the line  $l^{k-2}$ .

If the distance  $d$  is below the threshold we continue by constructing a new best fit line. This could be done by applying a linear best fit to all the points between  $p_0^k$  and  $\lambda_k$ , but it is faster to approximate the best fit line  $l^k$  as the line through  $p_0^k$  and  $p_1^k$  where

$$p_0^k = p_0^{k-1} - \frac{[(l_2^k)^\perp \cdot (\lambda^k - p_0^{k-1})] (l_2^k)^\perp}{2} \quad (12)$$

and

$$p_1^k = \lambda^k + \frac{[(l_2^k)^\perp \cdot (\lambda^k - p_0^{k-1})] (l_2^k)^\perp}{2} \quad (13)$$

This process is then repeated until a corner or discontinuity is found or the scan points are exhausted. When a corner is found the corner is set as the new start point  $p_0$  and a new line is started. When a discontinuity is found the start point is taken to be the next point after the discontinuity and a new line is started.

Pseudocode describing the method of line extraction is given in Algorithm 1 for the case where the obstacles are given in geometric order.

---

**Algorithm 1** Line Feature Extraction

---

```

1:  $d\alpha :=$  sensor angular resolution
2: for each obstacle in  $\Lambda$  do
3:   if  $\|d_i - d_{i+1}\| > (\sin(d\alpha) * \max([d_i d_{i+1}]))$  then
4:     save  $d_i$  as a discontinuity
5:   end if
6: end for
7:  $k :=$  look ahead distance
8: for each obstacle  $\lambda^k$  in  $\Lambda$  do
9:    $p_0 :=$  last detected corner
10:  if  $\lambda^{k+2}$  past next discontinuity then
11:     $p_1 :=$  next discontinuity
12:  else
13:     $p_1 := \lambda^{k+2}$ 
14:  end if
15:   $d := \|p_1 - p_0\|^\perp \cdot (p_1 - \lambda^k)$ 
16:  if  $d > \text{threshold}$  then
17:    fit a line through  $p_0$  and  $\lambda^k$ 
18:    take  $\lambda^k$  as the last obstacle
19:  end if
20: end for
```

---

#### IV.A.2. Exploring the Free Space

CSE operates directly on the output of a bearing and range sensor such as a scanning laser rangefinder. It requires only the location and heading of the vehicle and a representation of the obstacles present locally.

The starting circle is centered on the vehicle and expands to be tangent to the closest obstacle. We will represent this with a moving “expansion point”  $c \rightarrow \mathbb{R}^2$ . Each circle will be expanded from this point sequentially. The center of the starting circle center  $c_0$  is located at  $\hat{x}$ . This circle has a radius of

$$r_1 = \min_{0 \leq i \leq |\Lambda|} \|\lambda_i - \hat{x}\| \quad (14)$$

The first expansion will be perpendicular to the forward direction of the vehicle, at a distance proportional to the radius of the first circle.

$$c_i = c_{i-1} + (r_{i-1} - r_{min})[\cos \psi, \sin \psi]^T \quad (15)$$

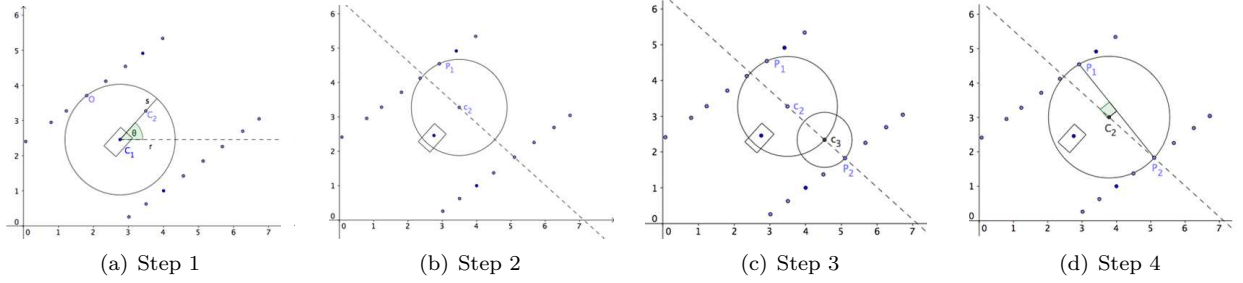
where  $r_{min}$  is a predefined minimum safe radius, usually taken as the radius of the smallest circle that would fully contain the vehicle, and  $\psi$  is the vehicle's heading.

The first expansion's forward direction is taken to be that of the vehicle.

$$\vec{f}_1 = [\cos \psi, \sin \psi]^T \quad (16)$$

After the location of the expansion point is found, the first circle expansion can begin. Figure 5 shows the sequence of expansions starting from the vehicle center through to the first fully expanded circle. The circle's center is constrained to a line perpendicular to the expansion forward direction which has the equation

$$0 = c_i + \beta \vec{f}_{i-1}; \beta \in \mathbb{R} \quad (17)$$



**Figure 5. The four-step circle expansion process starting from the vehicle location.**

The closest point to the expansion point is defined as

$$P_1 \equiv \min_{0 \leq i \leq |\Lambda|} \|\lambda_i - c_i\| \quad (18)$$

The expansion point is then relocated to the point lying on both the circle and the line in the direction away from the obstacle.

$$c_{i+1} = c_i + \{(c_i - P_1) \cdot \hat{l}\} \hat{l} \quad (19)$$

This expansion sequence is repeated iteratively until the closest obstacle to the expansion point  $c_i$  is not  $P_1$ . The obstacle that causes this condition is then defined as  $P_2$ . The center of the fully expanded circle can then be found as

$$C_{i+1} = (\overline{P_2 P_1})^\perp \cap \hat{l} \quad (20)$$

and the radius is

$$r_{i+1} = \|C_{i+1} - P_1\| \quad (21)$$

The expansion is then repeated until some termination condition is reached.

After a circle has been expanded it is necessary to pick a new forward direction to continue the expansion. The logical choice for a “safest-path” expansion is to expand away from both previously detected obstacles  $P_1$  and  $P_2$ . The safest-path expansion is defined as

$$\vec{f}_{i+1} = \{\vec{f}_i \cdot (P_2 - P_1)^\perp\} (P_2 - P_1)^\perp \quad (22)$$

This works well in most situations, however it can be advantageous to limit the rate of change of the forward direction to better represent the vehicle dynamics or limit erratic behavior in complex environments. Pseudocode for the circle expansion process is given in Algorithm 2.



---

**Algorithm 2** Circle Expansion Process

---

```
1: //  $c^k$  is the center of the last expansion
2: for  $\lambda_i$  in  $\Lambda$  do
3:    $r_i := \|\lambda_i - c^k\|$ 
4: end for
5:  $j := \operatorname{argmin}(r)$ 
6:  $r_0 := r_j$ 
7:  $p_0 := \lambda_j$ 
8:  $f :=$  forward direction of vehicle
9:  $c_1 := (r_0 - r_{safety})f$ 
10:  $l_1 := p_0 - c^k$ 
11:  $p_1 := \operatorname{closestObstacle}(c_1)$ 
12: while  $p_2 = p_1$  do
13:    $v_1 := (l_1 \cdot (c_k - p_0))l_1$ 
14:    $v_2 := (\|c^k - p_1\| - r_{safety})\frac{v_1}{\|v_1\|}$ 
15:    $p_2 := \operatorname{closestObstacle}(c^k + v_2)$ 
16: end while
17: // circle is bounded by  $p_1$ ,  $p_2$  and  $l_1$ 
```

---

#### IV.A.3. Terminating the Expansion

Usually for local online planning using CSE it is sufficient to stop after a fixed number of expansions or after a certain distance has been covered because local sensor measurements out past that range are unreliable. However a more complete analysis for offline planning would involve limiting the circle radii. If a circle cannot be expanded to the minimum radius a dead-end has been reached. If the circle should exceed some set maximum radius, there is a high likelihood that an open area has been detected. In either of these cases the circle expansion can be stopped. When planning a global path to a goal by using the entire map as generated up to that point circle expansion may be allowed to continue until the goal is reach or all likely paths have been exhausted. There is not cost penalty related to the number of circles expanded, beyond the linear (with respect to the number of features) cost of expanding each circle. The process of performing circle expansions is equivalent to the generation of a binary tree through the environment. Therefore, heuristic search methods such as A\* search may yield quicker convergence to the goal state <sup>18</sup> by minimizing the number of expansions performed.

With CSE it is possible to detect when a path has resulted in a dead end by simply observing when it is no longer possible to expand a circle of radius greater than half the width. [11] gives a method of then planning a path that will turn around and continue exploration of the environment.

## IV.B. Edge Cutting Tree

Suppose that, after processing raw data from range finder with a feature detection algorithm, we have a map of obstacles that are represented not only by points but also by lines. Such representation is computationally economical in indoor environments. To fully benefit from it, we propose a new path planning algorithm that is suitable for rover navigating through indoor environments.

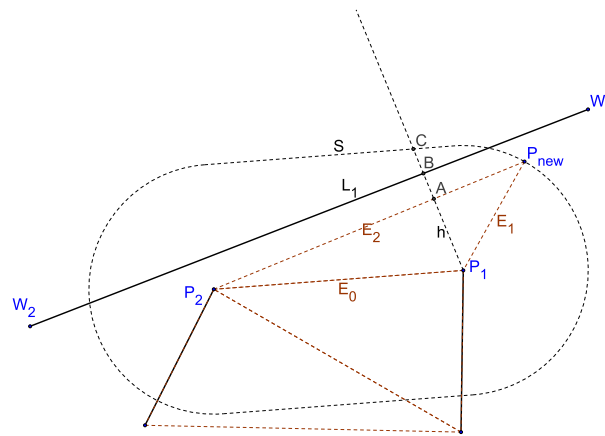
### IV.B.1. Basic Idea

The proposed algorithm performs an incremental triangulation over the points on the map, which could be end points of lines or nonlinear obstacles of arbitrary shape. Since each triangle is potentially a branching point of the path, the triangulation result is stored as a binary tree. From the binary tree, multiple paths can be identified by cutting through the edges of the triangles. In fact, given a triangulation of points on a 2D map, one can start from any edge to advance either to the left or to the right edge repeatedly unless it is part of a line obstacle or shorter than a threshold. From every edge, the next triangle is selected by recruiting a new point in front of the edge, which is done by checking the closest point from the edge. Fig.

12 (a) shows an example of triangulation, which shows the overlay of all the possible triangulations from a fixed starting edge. In this way, one can find paths that avoid obstacles on the map whether they are static or moving. A simple way for the vehicle to use the result of triangulation as its path is referring the vehicle to way points on the path. The crossing of a path with an edge is used as a way point for the vehicle. We also show that given a cost function to be minimized the way point position on each edge can be optimized.

#### IV.B.2. Special Cases

In addition to the basic description of the algorithm, there are certain cases when we need to insert an edge to the tree before advancing to the next triangle. Suppose we are currently at edge  $E_0$  and about to advance to  $E_1$  in Fig.6 if the foot  $A$  of the perpendicular line  $h$  from  $P_1$  toward  $E_2$  is found on  $E_2$ , then an additional edge will be inserted as a node in the tree before  $E_1$ . The edge to be inserted will be the shortest of the line segments between  $P_1$  and the following points:  $A$  (if  $E_2$  is a wall),  $B$ , or  $C$ . Point  $C$  is the intersection of  $h$  and a equidistant curve  $S$  from  $E_0$  through  $P_{new}$ . If there is a line obstacle  $L_1$  between  $C$  and  $P_1$ , then  $B$  should be chosen for the edge to be inserted instead of  $C$ . This action improves clearance of the path away from  $P_1$ .



**Figure 6.** ECT recruiting a new point and two edges to expand the tree: when the base angle of the triangle at  $P_1$  is large enough for  $A$  to exist on  $E_2$ , additional edge will be inserted between  $E_0$  and  $E_1$ . The edge will be the shortest of the line segments between  $P_1$  and  $A$  (if  $E_2$  is a wall),  $B$ , or  $C$ . This action helps ensuring the clearance of the path from obstacles by checking the length of the edge inserted.

#### IV.B.3. Properties

One can determine if each path is terminated as a dead end, an open end, or a loop. If both  $E_1$  and  $E_2$  are walls, or  $E_0$  is open but shorter than a threshold width, then it is supposed as a dead end. If there is no further point to recruit from the current edge, then it is regarded as an open end. If an edge was previously visited and if the direction of advance also is the same as the previous visit, then it is regarded as a loop.

The terminating condition can be used to narrow down the choice of the paths appropriate for the application. If the application requires the vehicle to proceed without stopping at blocked corners or without circling around an object, then it is obvious that the open-end paths must be chosen. This selection can substantially reduce the number of possible paths. From the open-end paths chosen, one can either randomly choose a path or, if target position or direction is known, deliberately choose a path that leads to the target possibly with the shortest distance to travel. Again depending on the purpose of application, one may want to choose a path so that the sum of the areas of the visited triangles is the largest if the mission is to cover a certain area. Once a reasonable number of paths are chosen, one can perform optimization of the way point position on each edge, for example, to minimize overall curvature.

A singularity of the algorithm is that when there is no point or line at all in the map, then it can not find a point to draw a triangle, and thus can not find any path even though it is facing a large open space. This issue should be considered when generating a map from any sensor data. For example, one can place points at virtual boundaries when the scan rays do not hit any objects.

#### IV.B.4. Pseudo Code

The pseudo code for the ECT algorithm is given in Algorithm 3.

---

#### Algorithm 3 Edge Cutting Tree

---

```

1:  $(t, T) = \text{tri}(P_1, P_2, T, W, V)$ 
2:  $// P_1, P_2$ : right and left end points of the current edge  $E_0$ 
3:  $// W$ : walls,  $V$ : visited edge
4:  $// N_{fwd}$  = Number of points in the front (visible or not)
5: if Visited( $P_1, P_2, V$ ) then
6:   Loop Detected; Store Current Edge
7: else if  $E_0 < E_{min}$  or  $E_0 \in W$  then
8:    $t = \text{DeadEnd}$ 
9: else
10:   $P_{new}$  = the closest AND visible point in the front
11:  if  $!P_{new}$  and  $N_{fwd} > 0$  then
12:     $t_l = \text{DeadEnd}$ ;  $t_r = \text{DeadEnd}$ 
13:  else if  $!(P_{new})$  and  $N_{fwd} = 0$  then
14:     $t_l = \text{OpenEnd}$ ;  $t_r = \text{OpenEnd}$ 
15:  else
16:    Add  $E_0$  to  $V$ 
17:    Find  $E_1^*$  to be inserted before  $E_1$ 
18:    if Need  $E_1^*$  and  $E_1^* < E_{min}$  then
19:       $t_r = \text{DeadEnd}$ 
20:    else
21:       $(t_r, T) = \text{tri}(P_1, P_{new}, T, W, V)$ 
22:      if Need  $E_1^*$  then
23:        Append  $P_1 P_2^*$  to  $T$ 
24:      end if
25:    end if
26:    Find  $E_2^*$  to be inserted before  $E_2$ 
27:    if Need  $E_2^*$  and  $E_2^* < E_{min}$  then
28:       $t_l = \text{DeadEnd}$ 
29:    else
30:       $(t_l, T) = \text{tri}(P_{new}, P_2, T, W, V)$ 
31:      if Need  $E_2^*$  then
32:        Append  $P_1^* P_2$  to  $T$ 
33:      end if
34:    end if
35:  end if
36:  Append  $E_0$  to  $T$ 
37: end if

```

---

#### IV.B.5. Consideration for On-line Implementation

Computation time is a critical factor in using the algorithm for real time application. General efforts to reduce computation time can be considered. Since visibility is checked in the algorithm when recruiting a new point from the current edge, using the sweep line algorithm<sup>19</sup> can reduce the computational effort from  $\mathcal{O}(n^2)$  to  $\mathcal{O}(n \log n)$ . Also, if the map purely consists of lines without clusters of points, it performs best. If one is using SLAM when navigating the vehicle, registering the triangles as objects in SLAM method can also save much resource because the triangulation can be done additionally each frame without starting from scratch.

## IV.C. Cylindrical Sector Expansion

Being limited to motion within a plane is a serious limitation for aerial vehicles which can limit their usefulness. To counter this limitation, CSE can be utilized in a quasi-2D manner, which allows altitude adjustments in response to obstacles in the environment. This is done by performing an additional expansion of each circle sector to get cylinder sectors.

### IV.C.1. Expanding in the Z-Direction

Cylindrical Sector Expansion is an extension of 2D CSE. With the obstacles segmented by altitude, each circle is allowed to grow up and down from the nominal altitude until an obstacle obstructs the expansion. This then gives the extents to which the vehicle can make altitude changes at this location in the environment.

Once a circle has been fully expanded within the plane located at the vehicle's altitude, it can be expanded according to Algorithm 4

---

#### Algorithm 4 Cylinder Expansion

---

```

1:  $z_{min}^k$  = absolute lower cylinder bounds
2:  $z_{max}^k$  = absolute upper cylinder bounds
3: for  $\lambda$  in  $\Lambda$  do
4:   if  $\sqrt{(\lambda_x - \hat{x}_x)^2 + (\lambda_y - \hat{x}_y)^2} < \text{current circle radius}$  then
5:     if  $\lambda_z < z_{min}^k$  then
6:        $z_{min}^k = \lambda_z$ 
7:     else if  $\lambda_z > z_{max}^k$  then
8:        $z_{max}^k = \lambda_z$ 
9:     end if
10:  end if
11: end for
12: for each altitude discretization between  $z_{min}^k$  and  $z_{max}^k$  do
13:   if  $z_{min}^k \leq z_{nominal} \leq z_{max}^{k-1}$  and  $z_{current} \neq z_{nominal}$  then
14:     terminate expansion
15:   else
16:     continue expansion
17:   end if
18: end for

```

---

### IV.C.2. Limitations

Figure 7 shows a simple test case in which the quasi-2D method is inadequate. The expansion would begin at the entrance to the room, and inside the room a fully expanded circle would fill the room. This circle would be bounded at the top by the roof and at the bottom by the floor. Because the cylinder would be significantly larger than the exit in the roof, it would be impossible to find the exit from the room. This example demonstrates the significant limitations to the quasi two-dimensional pathfinding in enclosed environments. Proper resolution of a path in this environment requires full 3D pathfinding.

## IV.D. Spherical Sector Expansion

Performing the free space explorations in unrestricted three-dimensional space has significant advantages over two-dimensional and quasi two-dimensional methods. With a full three-dimensional exploration method it is possible to explore every possible safe path, allowing the vehicle to consider parts of the environment that would otherwise be inaccessible. In this section we will be presenting a method for finding paths in a 3D environment. The method is an extension of Circular Sector Expansion in three dimensions. Whereas CSE will produce a binary tree consisting of the paths in a plane equidistant to all obstacles, SSE produces a

ternary tree of paths which are centered between opposing sets of obstacles. However, the extra dimensionality of SSE creates a key difference with CSE. While CSE approximates the edges of the Generalized Voronoi Diagram, SSE does not estimate the faces of a 3D Voronoi Diagram.

#### IV.D.1. Spherical Expansion

The expansion of each sphere starts in the same manner as CSE as described in .<sup>12</sup> The expansion is started from the current location of the vehicle, and the first constraint point  $p_0$  is taken as the closest obstacle in the environment to the vehicle. The initial circle then has a center at  $\hat{x}$  and a radius of

$$r_{c_0} = \min_{0 \leq i \leq |\Lambda|} \|\lambda_i - \hat{x}\| \quad (23)$$

The point which terminates this first stage of the expansion is denoted  $p_0$ . From here the expansion continues in a similar method to [12] by first fully expanding a circle, and then moving into the 3D realm to expand the circle into a sphere. The next constraint point is found by taking the intersection of the line connecting  $p_0$  and  $c_0$  with the perpendicular bisector plane between each obstacle and  $p_0$ . The process of performing a sphere expansion is shown in Algorithm 5. The sphere expansion is performed in sequence along different “slide lines”. The slide line is the direction along which the sphere expansion is constrained.

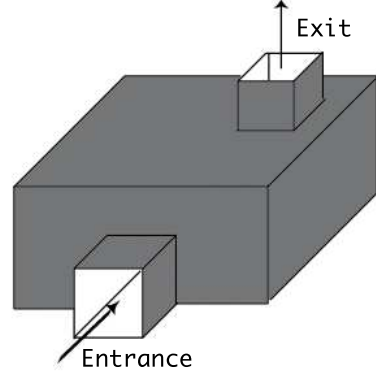


Figure 7. A simple example of a 2.5D failure case.

The initial slide line is defined by the expansion starting point, which is the location of the vehicle for the first circle expansion. Expanding the sphere along this line will give the next constraint point  $p_1$ . The sphere is now constrained by two obstacles  $p_0$  and  $p_1$ . The next expansion would only be bounded in one dimension and should be allowed to continue along the bisector plane between  $p_0$  and  $p_1$ . To constrain the expansion the next slide line is taken to be the projection of the forward direction of the vehicle on this plane. This gives the third constraint point  $p_2$ . The final constraint point is found by allowing the sphere to expand along the line which is perpendicular to the plane defined by the three constraint points  $p_0$ ,  $p_1$  and  $p_2$  and coincident with the center of the circle defined by the same points. After this expansion  $p_4$  will be known and the circle will be fully expanded.

#### IV.D.2. Developing the Search Tree

The four constraint points now define a tetrahedron, as they cannot all be coplanar. Each face of the tetrahedron is possible avenue for further expansion of the path. However one face of the tetrahedron is always going to face backwards towards the direction from which the path came, and therefore can be neglected when continuing the expansion. This leaves the path with three branches at each expansion. Therefore the expansion will be arranged as a trinary tree.

By using lazy evaluation when searching the tree for convergence to a goal state it is possible to eliminate the need to evaluate every branch of the tree out to the search depth. By assigning a cost function to evaluate each node of path as it is calculated, the entire tree can be searched using a graph search algorithm such as Dijkstra’s algorithm or A\*. Because these algorithms continually evaluate the lowest cost option, if they are applied as the tree is being created high cost paths can be eliminated early on.

## V. Experimental Results

We present results of experimentation and simulation in this section.

---

**Algorithm 5** Sphere Expansion

---

```
1:  $p_0 :=$  sphere constraint point
2:  $s_0 :=$  point in expansion direction
3: for each  $\lambda_i$  in  $\Lambda$  do
4:    $l_1 = p_0 - \lambda_i$ 
5:    $s_1 = \lambda_i + \frac{1}{2}l_1$ 
6:    $r_1 =$  any point not on  $l_1$ 
7:    $n_1 = s_1 \times r_1$ 
8:    $s_2 = s_1 + \hat{n}_1$ 
9:    $n_2 = n_1 \times l_1$ 
10:   $s_3 = s_1 + \hat{n}_2$ 
11:   $A = \begin{pmatrix} s_{0_x} - c_{0_x} & s_{2_x} - s_{1_x} & s_{3_x} - s_{1_x} \\ s_{0_y} - c_{0_y} & s_{2_y} - s_{1_y} & s_{3_y} - s_{1_y} \\ s_{0_z} - c_{0_z} & s_{2_z} - s_{1_z} & s_{3_z} - s_{1_z} \end{pmatrix}$ 
12:   $B = \begin{bmatrix} s_{0_x} - s_{1_x} \\ s_{0_y} - s_{1_y} \\ s_{0_z} - s_{1_z} \end{bmatrix}$ 
13:   $\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{A}{B}$ 
14:   $p_1 = s_0 + (c_0 - s_0)t$ 
15:   $d_i = \|p_1 - c_0\|$ 
16:  if  $d_i < d_{min}$  then
17:     $c_0^{k+1} = p_1$ 
18:     $r^{k+1} = \|\lambda_i - p_1\|$ 
19:  end if
20:  return  $c_0^{k+1}$  and  $r^{k+1}$ 
21: end for
```

---

## V.A. Guidance Control Command

First, we introduce a two-dimensional guidance control law that can efficiently make use of the safe paths generated from the path planning algorithms described in the previous section.

$$\tan \phi = -\frac{2L}{r_1} \sin \theta_1, \quad (24)$$

where  $\phi$  is the steering angle of the front wheel,  $L$  is the distance between front and rear wheels,  $r_1$  is the distance from the vehicle to the center of the first circle (if the Circular Section Expansion algorithm) or first way point (for the Edge Cutting Tree algorithm), and  $\theta_1$  is the initial heading error.

If we incorporate the second way point in the control law, the new guidance command is written as

$$\tan \phi = -\frac{2L}{d} \sin(\theta_1 + \alpha), \quad (25)$$

where

$$\begin{aligned} d^2 &= r_1^2 + r_2^2 + 2r_1r_2 \cos \theta_2 \\ \cos \alpha &= \frac{r_1^2 + d^2 - r_2^2}{2r_1d} \end{aligned} \quad (26)$$

## V.B. Heuristic Circular Sector Expansion

### V.B.1. 2D Test Platform

The vehicle test platforms are a series of modified Traxxas E-Maxx RC trucks. The vehicle's body has been replaced with a navigation system comprising a 1.6 Ghz mini-ITX computer, 5Ah lithium polymer battery and a 802.11g wireless card. The system has a SICK S300 scanning laser rangefinder attached to it, which communicates with the navigation computer over RS-422. The laser rangefinder provides distance measurements to obstacles in the surrounding 270 degrees in one-half degree increments. The measurements are provided at 12 Hz and reach to a distance of 30 metres.



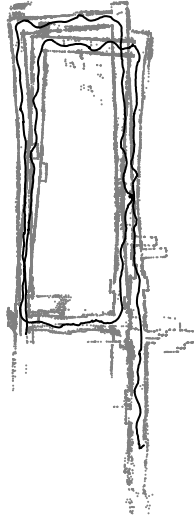
Figure 8. The vehicle test platform as equipped with a scanning laser rangefinder

The navigation computer runs an embedded linux distribution, and all processing is done through a custom message-based system which loads specific modules which pass important information amongst themselves in the form of messages. HCSE and the vehicle controller are implemented as separate modules, with the output of HCSE being passed to the controller.

### V.B.2. Test Results

Figure 10 shows a sequence of circle sector expansions as calculated on a pre-recorded drive segment. The vehicle was placed in a course which had been setup at the intersection of two hallways. It then drove around the oval track for several laps while collecting data from its laser rangefinder. This data was then used for offline testing of the CSE algorithm in MATLAB. The figure shows that the algorithm was quite capable in the area of short-term planning. Within the limits of the available data the algorithm did an excellent job of locating a feasible path.

The expansion sequence was limited to ten circles because it was observed that extending the sequence past that provided little in terms of useful results. With this limit in place the unoptimized code ran at approximately 30Hz against 540 obstacle points which were reduced before every iteration to eliminate excessively close points. When utilizing the line extraction methods outlined above it would be possible to plan paths against the entire map by performing CSE against the line features.



**Figure 9.** Drive through Howe Hall using HCSE as guidance method

Figure 9 shows an example of the full path as driven on the test rover running CSE as its sole method of motion planning. CSE was implemented as a C++ module which received the raw laser scans and outputted a series of expanded circles. The rover was instructed to drive a weighted average of the first two expanded circles. During the drive dynamic and static obstacles were presented to the rover, including people and other vehicles. The vehicle successfully managed to avoid all obstacles present, only getting stuck when the obstacles blocked off all feasible paths. By integrating a method of reversing the vehicle it would be possible to address these situations as well.

### V.B.3. Comparison with CSE

The primary advantage of HCSE over traditional CSE is the ability to work with features other than point features. However, other advantages can be observed when comparing the two methods. Figure 11 shows a typical expansion sequence using the standard CSE algorithm. Because Heuristic CSE uses a forward step size proportional to the radius of the expanded circle, fewer circles need to be expanded to fully explore the environment. This results in a smoother path for the vehicle controller as well as a reduced computation time.

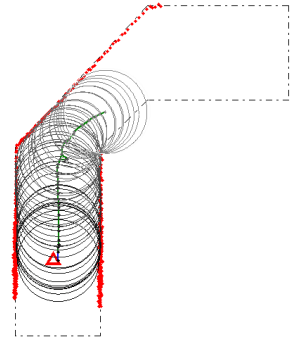
### V.C. Edge Cutting Tree

To demonstrate the feasibility of the algorithm, we used actual data stored from laser range finder running in a hallway. After rendering the data with line detection algorithm, we also added imaginary clustered points to represent obstacles. Fig.12 shows the map generated from one scan by the range finder and the paths with different terminating conditions as open end, dead end, and loop. Green lines mark the edge where the triangulation initiated on which the vehicle was actually located. The red x marks in Fig.12(a) represent mid points of the edges to travel through. In Fig.12(b) to (d), the solid lines are drawn by connecting way points on edges and it can be used as a reference path for the vehicle.

The way points are optimized with the following cost function with the gradient-descent method.

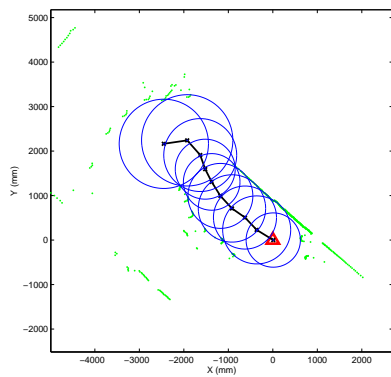
$$J = \sum_{i=2}^{n-1} (\cos(W_{i-1}W_iW_{i+1}) + 1),$$

where  $n$  is the number of way points  $W_i$ . The location of the starting point is fixed as a random point on the first edge and the last way point is fixed to be the mid point on the last edge. The optimization will try to enforce the angles to be close to 180 degrees to make the overall path as straight as possible to save the effort for turning. This allows the vehicle to steer away from the corner before reaching it, which is usually a better way than to drive straight toward the corner and make a stiff turn. When the cost function is more complicated, genetic algorithm can be considered for use. The optimization result can be used to verify if the path will be feasible given the minimum turning radius in advance. If the minimum turning radius is a function of linear velocity, the forward speed can be adjusted based on our on-line look-ahead path planning.

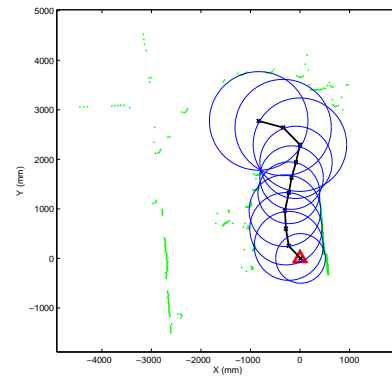


**Figure 11.** A typical expansion sequence of traditional CSE

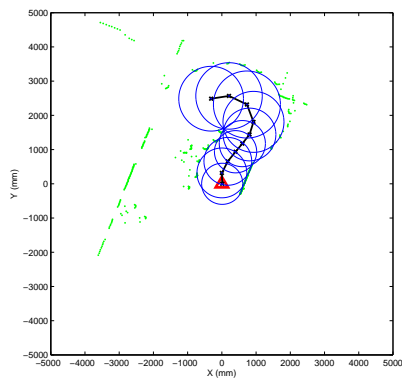




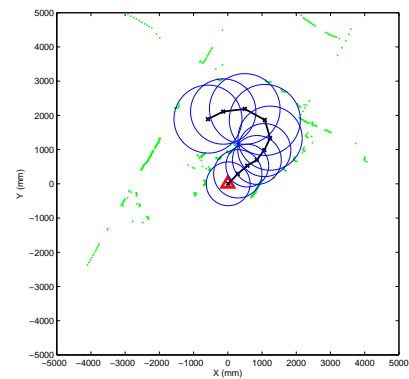
(a)  $t=0s$



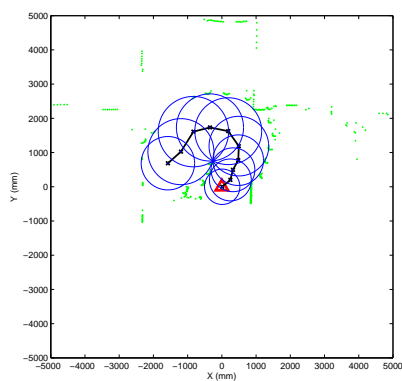
(b)



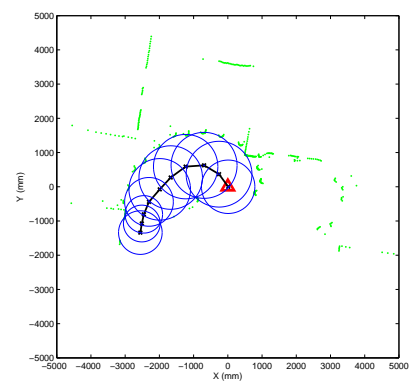
(c)



(d)

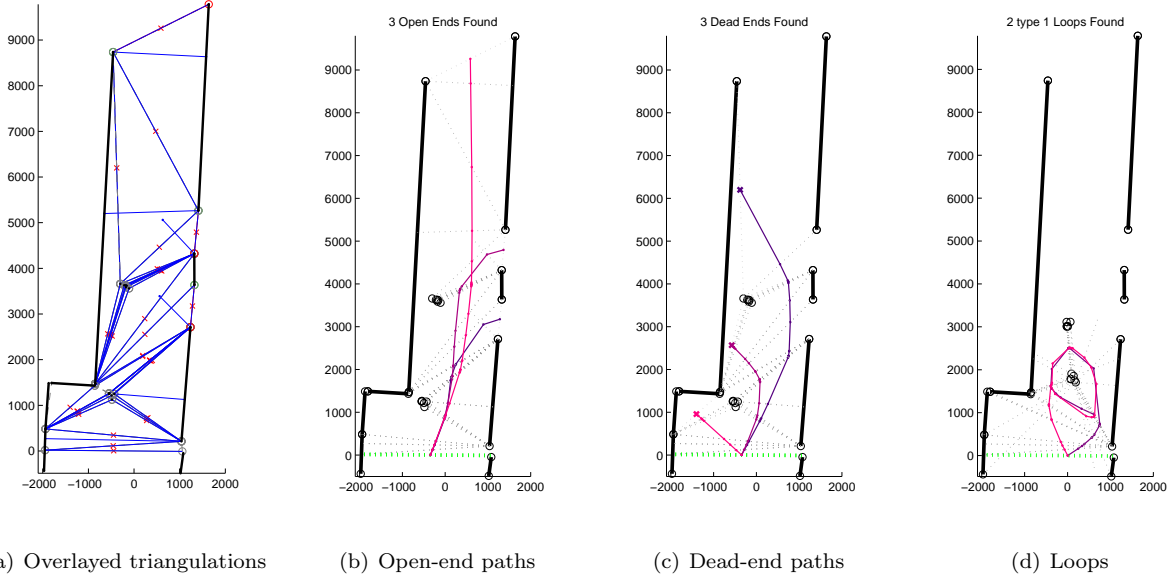


(e)



(f)

Figure 10. A series of circle expansions on a pre-recorded drive segment



**Figure 12.** The algorithm generated multiple paths, categorized them with ending conditions, and optimized the way point locations. Green lines are the starting edges on which a vehicle was sitting. Imaginary points are added to represent obstacles marked as black circles. (a) Overlaid triangulations for all the branches in the binary tree (b-c) Paths that end with open or closed space (d) Looping paths were found when the obstacles were moved.

## V.D. Spherical Sector Expansion

The SSE algorithm was tested in a MATLAB environment designed to simulate an indoor environment such as would be seen by a MAV. The test environment consisted of a 3D point cloud representing a hallway with a small opening at the end of one wall. SSE was used to plan a path from one end of the hall with the objective of escaping from the environment. The test was developed to verify that SSE was capable of planning paths in a full 3D environment. Figure 13 shows the results of the test run, with SSE successfully planning a path through the small opening at the end of the hallway.

The main path is visible in the series of large spheres that are centered in the hallway. Several side forks in the path are visible in small spheres. These paths were terminated due to the minimum radius of the next expansion being under the vehicle safety radius.

## VI. Conclusion

In this paper, we have introduced a set of new methods for path-planning in constrained unknown environments. The methods have encompassed various levels of complexity in environmental models, computational requirements and path capability. Heuristic Circular Sector Expansion (HCSE) was demonstrated through ground tests in complex static and dynamic environments and shown to be robust to disturbances and changing environments. This 2D method was then extended into the quasi-2D realm, which provided a method of incorporating altitude changes into the computed paths. Finally Spherical Sector Expansion (SSE) was introduced, which provided a method for pathfinding in all three dimensions. SSE was tested in simulation and the results were promising. Parallel to the circular and spherical sector based methods, we also presented Edge Cutting Tree (ECT), which exploits a map of obstacles that are represented not only by points but also by lines. Such representation is computationally economical in indoor environments. Results of experiments and simulation showed the effectiveness of the proposed approaches.

The methods presented in this paper provide a collection of alternatives for pathfinding in environments that are defined by an abundance of obstacles. These methods will be further explored through simulation and flight testing to verify their effectiveness and suitability for online path-planning.

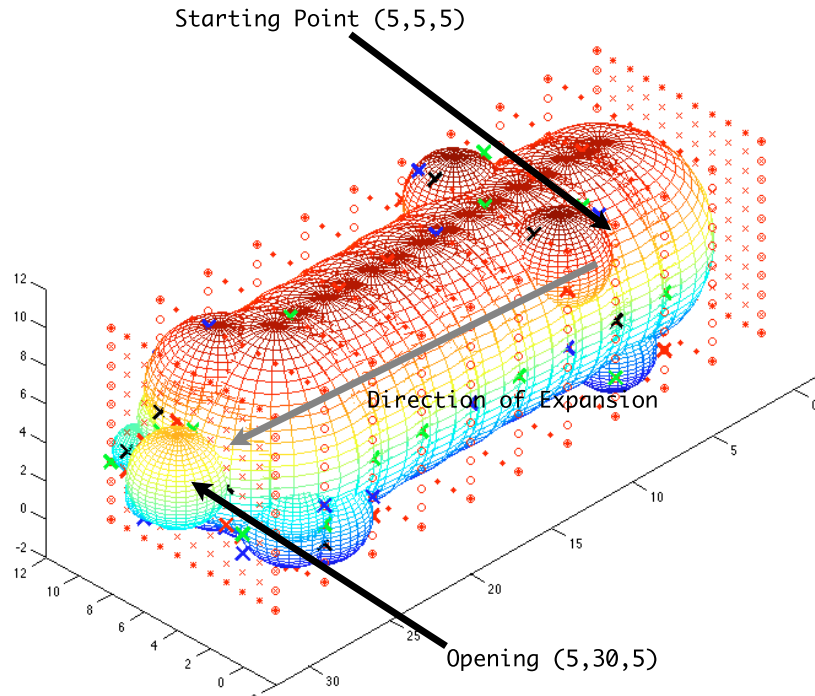


Figure 13. A demonstration of SSE algorithm in a simulated hallway with an opening at one end

## Acknowledgments

The research reported in this article was in part supported by the Air Force Office of Scientific Research (AFOSR) and the Information Infrastructure Institute (I3) at Iowa State University. The authors would like to express their gratitude to Prof. James Oliver and Prof. Arun Somani at Iowa State University for their technical and financial support.

## References

- <sup>1</sup>Nardi, R. D. and Holland, O., "SwarMAV: A Swarm of Miniature Aerial Vehicles," *Proceedings of the 21st Bristol UAV Systems Conference*, 2006, pp. 34.1–34.9.
- <sup>2</sup>Petti, S. and Rocquencourt, I., "Safe Motion Planning in Dynamic Environments," *In IEEE-RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 2210–2215.
- <sup>3</sup>Parunuk, H., Breukner, S., and Odell, J., "Swarming Coordination of Multiple UAV's for Collaborative Sensing," *In Proceedings of Second AIAA "Unmanned Unlimited" Systems Conference*, 2003.
- <sup>4</sup>Kim, B., Hubbard, P., and Necsulescu, D., "Swarming Unmanned Aerial Vehicles: Concept Development and Experimentation, A State of the Art," 2004.
- <sup>5</sup>Root, P. J., *Collaborative UAV Path Planning with Deceptive Strategies*, Ph.D. thesis, MIT, Massachusetts, US, 2005.
- <sup>6</sup>Nechyba, M. C., Ifju, P. G., and Waszak, M., "Towards Flight Autonomy: Vision-Based Horizon Detection for Micro Aerial Vehicles," *In Proceedings of Florida Conference on Recent Advances in Robotics*, 2002.
- <sup>7</sup>Oh, P. Y. and Green, W. E., "Neural Nets and optic flow for autonomous micro-air-vehicle navigation," *In Proceedings of the ASME International Mechanical Engineering Congress and Exposition*, 2004.
- <sup>8</sup>Green, W. E., Oh, P. Y., and Barrows, G., "Flying Insect Inspired Vision for Autonomous Aerial Robot Maneuvers in Near-Earth Environments," *IEEE International Conference on Robotics and Automation*, 2004, pp. 2347–2352.
- <sup>9</sup>Smith, P., Reid, I., and Davison, A., "Real-Time Monocular SLAM with Straight Lines," *British Machine Vision Conference*, Vol. 1, September 2006, pp. 17–26.
- <sup>10</sup>Berg-Taylor, K., Seo, K., and Chung, S.-J., "Development of a Car-like Online Navigation Testbed," *IEEE International Conference on Electro/Information Technology*, Ames, IA, Ames, IA, MAY 2008.
- <sup>11</sup>Ronnback, S., Berglund, T., and Freriksson, H., "On-Line exploration by Circle Sector Expansion," *Proceedings of the IEEE Conference on Robotics and Systems*, IEEE, Kunming, China, 2006.
- <sup>12</sup>Ronnback, S., *On Methods for Assistive Mobile Robots*, Ph.D. thesis, Lulea University of Technology, Sweden, June 2006.
- <sup>13</sup>Bhattacharya, P. and Gavrilova, M. L., "Roadmap-Based Path Planning using the Voronoi diagram for a Clearance-Based Shortest Path," *IEEE Robotics and Automation Magazine*, 2008, pp. 58–66.

- <sup>14</sup>LaValle, S. M., “Rapidly-Exploring Random Trees: A New Tool for Path Planning,” Tech. Rep. TR 98-11, Computer Science Department, Iowa State University, October 1998.
- <sup>15</sup>Frazzoli, E., *Robust Hybrid Control for Autonomous Vehicle Motion Planning*, Ph.D. thesis, Massachusetts Institute of Technology, 2001.
- <sup>16</sup>Vandapel, N., Kuffner, J., and Amidi, O., “Planning 3-D Path Networks in Unstructured Environments,” *In Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005.
- <sup>17</sup>Fischler, M. A. and Bolles, R. C., “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” 1987, pp. 726–740.
- <sup>18</sup>Dechter, R. and Pearl, J., “Generalized best-first search strategies and the optimality of A\*,” *J. ACM*, Vol. 32, No. 3, July 1985, pp. 505–536.
- <sup>19</sup>Shamos, M. and Hoey, D., “Geometric Intersection Problems,” *Proc. 17th Annu. IEEE Symp. Found. Computer Sci.*, 1976, pp. 208–215.